Adaptive reflex autonomicity for real-time systems

Innovations Syst Softw Eng (2009) 5:107–115
DOI 10.1007/s11334-009-0090-7

RESEARCH PAPER

Adaptive reflex autonomicity for real-time systems

Roy Sterritt · Mike Hinchey

Received: 15 March 2009 / Accepted: 20 April 2009 / Published online: 16 June 2009

Abstract It may appear that for software systems that require strict real-time behavior, the idea of incorporating self-management (and specifically concepts from Autonomic Computing) may add the burden of excessive additional functionality and overhead. However, our experience is that, not only does real-time software benefit from autonomicity, but also the Autonomic Computing initiative (like other initiatives aiming at self-management) requires the expertise of the real-time community in order to achieve its overarching vision. In particular, there are emerging classes of real-time systems for which incorporation of self-management is absolutely essential in order to implement all of the requirements of the system, and in particular the timing requirements.

1 Introduction

The term "Autonomic Computing" was coined by IBM in their call to industry in 2001 [2] (although 'Autonomic' in computing was initially used by DARPA under the Autonomic Information Assurance program in late 1990s [13]). The approach takes inspiration from the human and mammalian nervous systems in an attempt to develop systems that are self-managing and ultimately self-governing [1].

We have argued in a previous paper [3] that computer-based systems *should* be autonomic. We stand by that belief, in particular as software becomes more complex, while simultaneously more mobile, pervasive, embedded and ubiquitous, with greater expectations in terms of functionality, performance and real-time behavior. In this paper [3], we were addressing a particular audience, viz. members of the IEEE Technical Committee on Engineering Computer-Based Systems (TC-ECBS). To this community, "computer-based systems" typically means embedded systems involving both hardware and software.

Like real-time software systems, such systems often have strict constraints in terms of performance and timing requirements. Adding concepts from Autonomic Computing (AC) might seem, at a first glance, to *add* complexity rather than reduce it, clearly something that cannot be afforded in many real-time software systems. We believe, however, that Autonomic Computing has much to offer in terms of *reducing*, or at least systematically coping with, complexity, and our work over the last number of years has looked at how we might exploit AC and other biologically inspired techniques in classes of systems that would otherwise be infeasible.

We believe that Autonomic Computing is not achievable without real-time systems (in particular at lower, implementation levels). Simultaneously, we believe that real-time software systems can benefit much from AC, without significant overhead, and that future advancements necessitate that we move towards self-managing real-time software systems if we are to be able to meet the strict timing constraints of many new applications and environments.

2 Self-managing systems

A number of initiatives have emerged with the vision of achieving self-management in our computing and communications systems. These include Autonomic Computing and Autonomic Communications, and also Organic Computing,

R. Sterritt (✉)
University of Ulster, Jordanstown, Belfast, Northern Ireland, UK
e-mail: r.sterritt@ulster.ac.uk

M. Hinchey
Lero-the Irish Software Engineering Research Centre,
University of Limerick, Limerick, Ireland
e-mail: mike.hinchey@lero.ie

Adaptive Infrastructure, N1, Dynamic System Initiative, Adaptive Network Care, Proactive Computing, Self-Organizing Systems, Self-*, Selfware, Biologically-Inspired Computing [5].

Although these different initiatives may have different academic specialties at their base, or were initiated by different industrial organizations, often they are inspired by self-management as exhibited in biology and nature. The IEEE has formalized the community as a Technical Committee on "Autonomous and Autonomic Systems" to underscore the strategic aims of developing next generation self-organizing and self-managing computing and communications infrastructures and systems.

2.1 Autonomic computing

The Autonomic Computing (AC) initiative focuses on managing complexity with self-managing systems, taking inspiration from the human autonomic nervous system (ANS) [1–5].

The ANS is the part of the nervous system that controls the vegetative functions of the body, such as circulation of the blood, intestinal activity, and secretion and production of chemical "messengers" (hormones) that circulate in the blood. The sympathetic nervous system (SyNS) supports "fight or flight", providing various protection mechanisms to ensure the safety and well-being of the body. The parasympathetic nervous system (PaNS) supports "rest and digest", ensuring that the body performs necessary functions for long-term health.

The general properties of an autonomic (self-managing) system can be summarized by four objectives: being self-configuring, self-healing, self-optimizing and self-protecting, and four attributes: self-awareness, self-situated, self-monitoring and self-adjusting [7]. Essentially, the objectives represent broad system requirements, while the attributes identify basic implementation mechanisms.

Self-configuration represents a system's ability to re-adjust itself automatically; this may simply be in support of changing circumstances, or to assist in self-healing, self-optimization or self-protection.

Self-healing, in reactive mode, is a mechanism concerned with ensuring effective recovery when a fault occurs, identifying the fault, and then, where possible, repairing it. In proactive mode, it monitors vital signs in an attempt to predict and avoid "health" problems (reaching undesirable situations).

Self-optimization means that a system is aware of its ideal performance, can measure its current performance against that ideal, and has defined policies for attempting improvements. It may also react to policy changes within the system as indicated by the users. A self-protecting system will defend itself from accidental or malicious external attack. This necessitates awareness of potential threats and a means of handling those threats.

In achieving such self-managing objectives, a system must be aware of its internal state (self-aware) and current external operating conditions (self-situated). Changing circumstances are detected through self-monitoring and adaptations are made accordingly (self-adjusting).

As such, a system must have knowledge of its available resources, its components, their desired performance characteristics, their current status, and the status of interconnections with other systems, along with rules and policies for how these may be adjusted. Such ability to operate in a heterogeneous environment will require the use of open standards to enable global understanding and communication with other systems.

These mechanisms are not independent entities. For instance, if an attack is successful, this will necessitate self-healing actions, and a mix of self-configuration and self-optimization, in the first instance to ensure dependability and continued operation of the system, and later to increase self-protection against similar future attacks. Finally, these self-mechanisms should ensure that there is minimal disruption to users, avoiding significant delays in processing.

The architecture of Autonomic Systems (AS) essentially consists of cooperating autonomic elements made up of the component that is required to be managed, and the autonomic manager [11,12]. It is assumed that an autonomic manager (AM) is responsible for a managed component (MC) within a self-contained autonomic element (AE). This autonomic manager may be either designed as part of the component or provided externally to the component, as an agent for instance. To achieve self-management, AEs will cooperate with remote autonomic managers through virtual, peer-to-peer, client–server or grid configurations.

2.2 Aims of AC

In their 2001 call to industry, IBM set out eight overriding objectives [2]. To the real-time systems (RTS) researcher or practitioner, several of these will be characteristics that are common to real-time systems, or at least very desirable:

1. To be autonomic, a computing system needs to "know itself"—and comprise components that also possess a system identity.
2. An Autonomic Computing System must configure and reconfigure itself under varying and unpredictable conditions.
3. An Autonomic Computing System never settles for the status quo—it always looks for ways to optimize its workings.

4. An Autonomic Computing System must perform something akin to healing—it must be able to recover from routine and extraordinary events that might cause some of its parts to malfunction.
5. A virtual world is no less dangerous than the physical one, so an Autonomic Computing System must be an expert in self-protection.
6. An Autonomic Computing System knows its environment and the context surrounding its activity, and acts accordingly.
7. An Autonomic Computing System cannot exist in a hermetic environment.
8. *Perhaps* most critical for the user, an Autonomic Computing System will anticipate the optimized resources needed while keeping its complexity hidden.

This list of eight requirements, set out in [2] and [4], points to the fact that to qualify as an Autonomic System, a system has to have a degree of self-awareness and familiarization with the components that compose it. It must know their capabilities, if it is to be able to adapt to its environment successfully, analogous to the way that dynamic scheduling requires a knowledge of tasks, their periods and deadlines.

An AS must be able to adapt to unpredictable conditions and adjust itself as necessary. This is something that is done, almost routinely, in an RTS. Real-time systems are constantly evolving. Their role is to adapt to changing environmental circumstances or new inputs. The role of scheduling is to optimize the use of resources and to make efficient use of available processing power. An effective RTS is performing this role.

A move towards management and reduction of inherent complexity is a key focus of the list of essentials given above. Complexity is a major issue for any large-scale system. Real-time systems have the added difficulty of having to meet strict timing-constraints and inputs from many sources, all of which need to be met.

But surely adding self-managing abilities to an already complex system, such as exhibited by an RTS, is only *adding* to adherent complexity? The overhead of self-management essentially adds an overhead to any system. Many RTSs are already pushing the limits of their schedules and adding any additional overhead is impossible. So is it even feasible to talk of autonomic real-time systems?

It is our contention that sometimes adding this overhead is justified. Indeed, there are times when it may even be essential, in particular for particular classes of systems where self-management, and even self-government, are vital. We are thinking of systems for which timing constraints could simply not be met any other way and for which the overhead of AC is more than repaid by a corresponding reduction in complexity.

2.3 Implementing AC

Handing over control that was previously held by a human to the system itself, in principle will add additional functionality, complexity and processing overhead to the system. This fact will be obvious to any practitioner in the field. Experience from AI, Expert Systems, and Machine Learning problems has shown many how slow decision-making systems can be. As such, it is vital to develop key principles and engineering techniques for how self-managing systems should be created.

If we reconsider the meaning of the actual terms "Autonomic" and "Autonomous" [6]:

au·to·nom·ic (àwtə nómmik)
adj.

1. Physiology.
   a. Of, relating to, or controlled by the autonomic nervous system.
   b. Occurring involuntarily; automatic: *an autonomic reflex.*
2. Resulting from internal stimuli; spontaneous.

au·ton·o·mic·i·ty (àwtə nóm i síttee)
n.

1. The state of being autonomic.

au·ton·o·mous (aw tónnəməs)
adj.

1. Not controlled by others or by outside forces; independent: *an autonomous judiciary; an autonomous division of a corporate conglomerate.*
2. Independent in mind or judgment; self-directed.
3. a. Independent of the laws of another state or government; self-governing.
   b. Of or relating to a self-governing entity: *an autonomous legislature.*
   c. Self-governing with respect to local or internal affairs: *an autonomous region of a country.*
4. Autonomic.

[From Greek autonomos: auto-, *auto-* + nomos, *law*]

We note a difference in their intent: "Autonomic" is very much concerned with spontaneous, reflex reactions while "Autonomous" is a slower, high-level conscious decision-making process.

The basic principles of Autonomic and Autonomous systems can be incorporated into the design of a system to ensure that the correct response rate is achieved where it is needed. This has resulted in us considering a simple

three-tiered abstract architecture in our designs of self-managing systems:

- Autonomous layer
- Selfware layer
- Autonomic layer

The *Autonomic layer* is the bottom tier, closest to the hardware, and operates with immediate reaction to situations to ensure that system operations are maintained.

The *Selfware layer* incorporates day-to-day operations of self-managing activity as and when needed, and as and when the system has the processing capacity available.

The *Autonomous layer* is the top tier where high-level strategic objectives of the system are directed and satisfied over time. This often includes reflection.

As such, a key element that we have included in our work in designing AS is that of the *Autonomic Reflex*, borrowed from embedded and real-time systems and extended to include active system health telemetry.

2.4 Autonomic reflex reactions: pulse monitoring

Essentially, the aim of Autonomic Computing is to create robust, dependable self-managing systems [8] in an attempt to deal with complexity.

At the heart of the architecture of any autonomic system are sensors and effectors. A control loop is created by monitoring behavior through sensors, comparing this with expectations (knowledge, as in historical and current data, rules and beliefs), planning what action is necessary (if any), and then executing that action through effectors. The closed loop of feedback control provides the basic backbone structure for each system component. There are two conceptual control loops in an Autonomic Element—one for self-awareness and another for self-situation (environmental awareness and context-awareness).

IBM represents this self-monitor/self-adjust control loop as the monitor, analyze, plan and execute (MAPE) control loop. The monitor-and-analyze parts of the structure process information from the sensors to provide both self-awareness and an awareness of the external environment. The plan-and-execute parts decide on the necessary self-management behavior that will be executed through the effectors. The MAPE components use the correlations, rules, beliefs, expectations, histories, and other information known to the autonomic element, or available to it through the knowledge repository within the Autonomic Manager (AM).

The autonomic environment requires that autonomic elements and, in particular, autonomic managers communicate with one another concerning self-*activities, in order to ensure the robustness of the environment [9,10]. It is our belief that the AM communications must also include a reflex signal.

To facilitate this, fault-tolerant mechanisms such as a heart-beat monitor ('I am alive' signals) and pulse monitor (urgency/reflex signals) may be included within the autonomic element [9,10]. See Fig. 1 for an illustration of our autonomic environment.

The notion behind the pulse monitor (PBM) is to provide an early warning of an undesirable condition so that preparations can be made to handle the processing load of diagnosis and planning a response, including diversion of load. Together with other forms of communications it creates dynamics of autonomic responses [11]—the introduction of multiple loops of control, some slow and precise, others fast and possibly imprecise, fitting with the biological metaphors of reflex and healing [9].

2.5 Reducing the monitoring workload through collaboration

This reflex component (pulse monitor) may be used to safeguard the autonomic element by communicating its health to another AE in a peer-to-peer fashion (Figs. 1, 2). The component may also be utilized to communicate environmental health information—i.e. how the AE perceived the health of the environment at that moment in time.

To assist in realizing real-time requirements and in reducing the self-managing monitoring burden on all elements, the pulse monitor can be utilized in a form of 'neighbourhood watch scheme' [10]. For instance, in the situation where each PC in a LAN is equipped with an autonomic manager, rather than each of the individual PCs monitoring the same environment, a few PCs (likely the least busy machines) may take on this role and alert the others via a change in pulse to indicate changing circumstances (Fig. 3).

An important aspect concerning the reflex reaction and the pulse monitor is the minimization of data sent—essentially only a "signal" is transmitted. Strictly speaking, this is not mandatory; more information may be sent, yet the additional information must not compromise the reflex reaction and the required real-time response of the system. For instance, in the absence of bandwidth concerns, information that can be acted upon quickly and will not incur processing delays can be sent. The important aspect is that the information must be in a form that can be acted upon immediately and does not involve processing delays (such as is the case of event correlation).

Just as the beat of the heart has a double beat (lub–dub) the autonomic element's pulse monitor may have a double beat encoded as described above: a self-health/urgency measure and an environment health/urgency measure. These match directly with the two conceptual control loops within the AE, and the self-awareness and environment awareness (self-situation) properties.

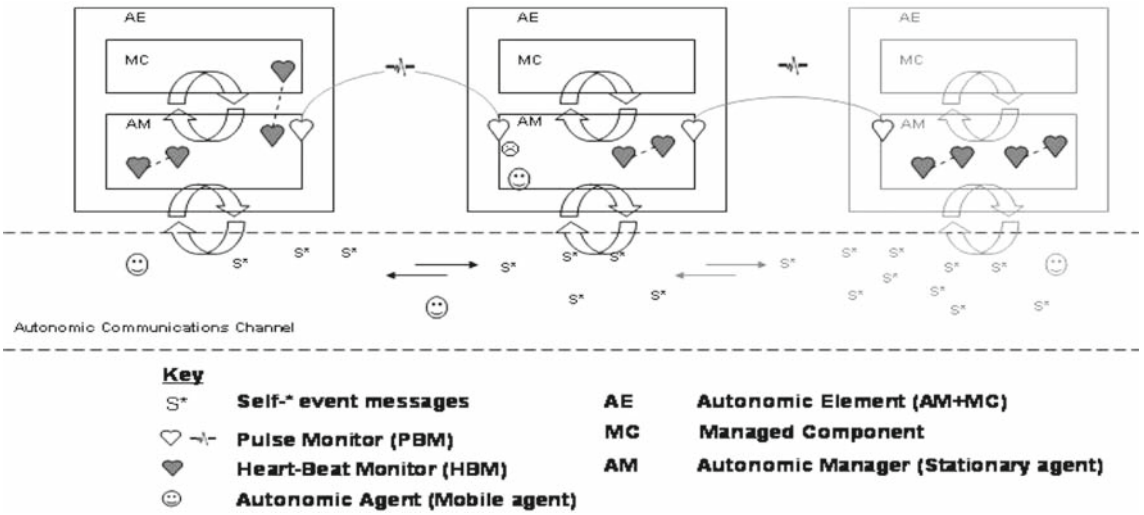

Fig. 1 An autonomic environment

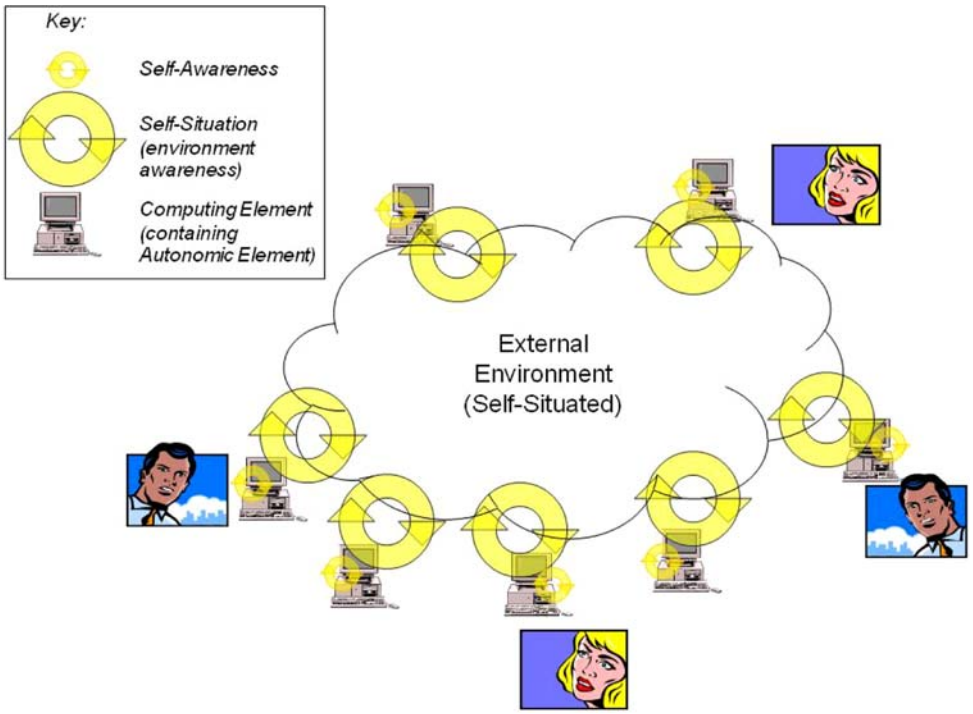

Fig. 2 An example autonomic environment—LAN two conceptual control loops—self awareness and environment awareness

2.6 Adaptive pulse monitoring (PaNS mode)

The standard heart-beat monitor (HBM) sends an 'I am alive' signal at constant static intervals. The extended version of this, the Pulse monitor [8,9], encodes within its signal not only a heartbeat but an 'I am healthy/unhealthy' signal in reflex reaction to a change in vital signs. Under normal conditions the pulse would act like the HBM, sending at regular intervals, yet on encountering circumstances affecting the system the pulse rate will increase to warn of the problem. This dynamic pulse rate is consistent with the biological metaphor, but it is also desirable to ensure that information is reported more frequently when operating conditions become difficult (flight or fight, SyNS). To achieve the reflex reaction a signal should be sent immediately, implying a change in the pulse rate, which should then stay high, reporting state information, until the situation is resolved.

Yet we are very aware of scenarios, for instance in the telecommunications domain [17], where under fault conditions a major fault can cause such a cascade of alarm event messages that it affects the real-time operation of the system and appears non-deterministic; under such management event message flooding it can be difficult to even provide adequate service [17].

Since the management event data flooding under fault conditions can add to the degradation of the system, we

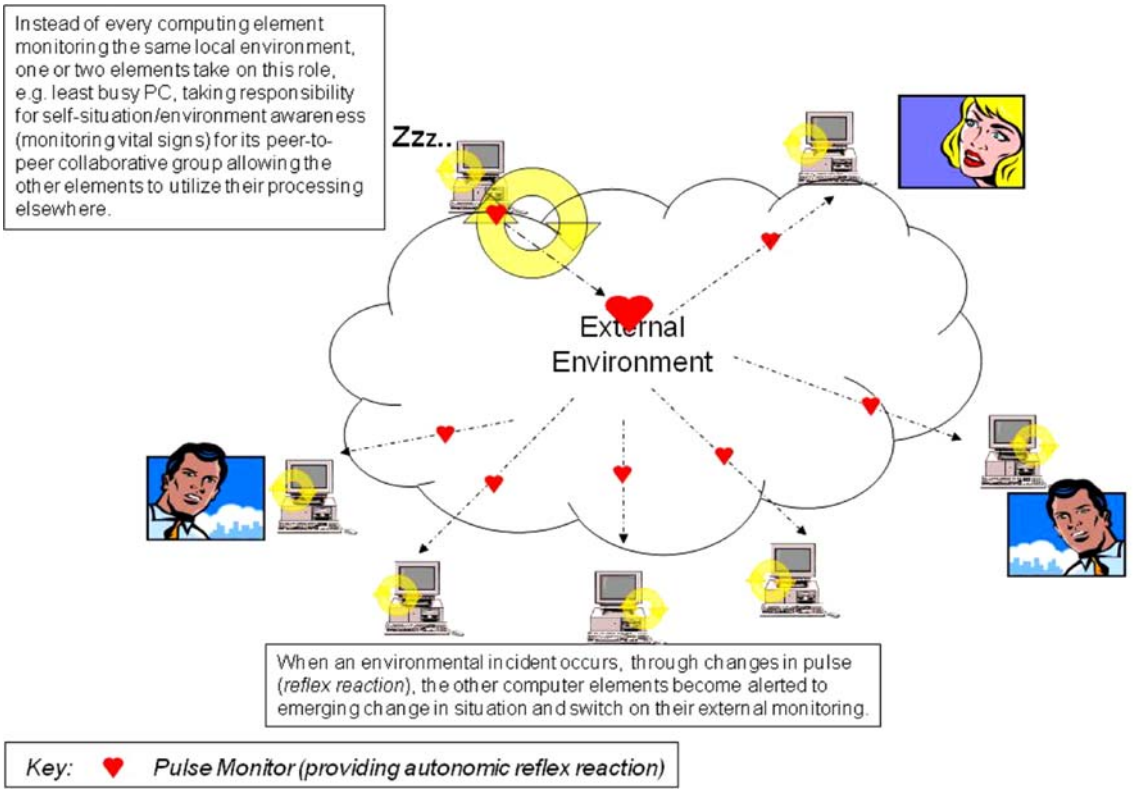

Fig. 3 Pervasive Autonomic Computing Monitoring Environment (PACMEn)—assisting with real-time constraints by reducing the monitoring burden through neighbourhood watch scheme

have been working on another extension, the adaptive pulse monitor, whereby the rate of the pulse will adapt to take into consideration bandwidth concerns and the congestion on the network. In effect, after the initial reflex reaction, the pulse and other self-*event messages would actually decrease. As such, upon detecting that the flood of management event messaging, along with limited bandwidth and resulting congestion, was adding to the system degradation, a specific pulse signal would be sent to reduce the necessity of sending the autonomic messages until the situation is resolved (in effect putting the self-managing system into rest and digest, parasympathetic (PaNS) mode). The key concept is that we must actively reduce alerting so that achieving autonomicity is not actually making the situation and response worse.

3 Next generation self-managing RTS example

3.1 NASA missions

NASA missions require the use of complex hardware and software systems, and embedded systems, often with hard real-time requirements [3]. Most of the missions involve significant degrees of autonomous behavior, often over significant periods of time. There are missions which are intended only to survive for a short period, and others which will continue for decades, with periodic updates to both hardware and software. Some of these updates are pre-planned; others, such as with the Hubble Space Telescope, were not planned but have now been undertaken (with updates performed by astronauts and via a robotic arm).

While missions typically have human monitors, many missions involve very little human intervention, and then often only in extreme circumstances. It has been argued that NASA systems *should* be autonomic [9,14], and that all autonomous systems should be autonomic by necessity. Indeed, the trend is in that direction in forthcoming NASA missions.

We take as our example, a NASA concept mission, ANTS, which has been identified [15] as a prime example of an autonomic system.

3.1.1 ANTS

ANTS is a concept mission that involves the use of intelligent swarms of spacecraft. From a suitable point in space (called a Lagrangian), 1,000 small spacecrafts will be launched towards the asteroid belt.

As many as 60–70% of these will be destroyed immediately on reaching the asteroid belt. Those that survive will coordinate into groups, under the control of a leader, which

will make decisions for future investigations of particular asteroids based on the results returned to it by individual craft which are equipped with various types of instruments.

Self-configuring ANTS will continue to prospect thousands of asteroids per year with large but limited resources. It is estimated that there will be approximately 1 month of optimal science operations at each asteroid prospected. A full suite of scientific instruments will be deployed at each asteroid. ANTS resources will be configured and re-configured to support concurrent operations at hundreds of asteroids over a period of time.

The overall ANTS mission architecture calls for specialized spacecraft that support division of labor (rulers, messengers) and optimal operations by specialists (workers). A major feature of the architecture is support for cooperation among the spacecraft to achieve mission goals. The architecture supports swarm-level mission-directed behaviors, subswarm levels for regional coverage and resource-sharing, team/worker groups for coordinated science operations and individual autonomous behaviors. These organizational levels are not static but evolve and self-configure as the need arises. As asteroids of interest are identified, appropriate teams of spacecraft are configured to realize optimal science operations at the asteroids. When the science operations are completed, the team disperses for possible reconfiguration at another asteroid site. This process of configuring and reconfiguring continues throughout the life of the ANTS mission.

Reconfiguring may also be required as the result of a failure, such as the loss of, or damage to, a worker due to collision with an asteroid (in which case the role may be assumed by another worker, which will be allocated the task and resources of the original).

Self-healing ANTS is self-healing not only in that it can recover from mistakes, but self-healing in that it can recover from failure, including damage from outside forces. In the case of ANTS, these are non-malicious sources: collision with an asteroid, or another spacecraft, etc.

ANTS mission self-healing scenarios span the range from negligible to severe. A negligible example would be where an instrument is damaged due to a collision or malfunctioning. In such a scenario, the self-healing behavior would be the simple action of deleting the instrument from the list of functioning instruments. A severe example would arise when the team loses so many workers it can no longer conduct science operations. In this case, the self-healing behavior would include advising the mission control center and requesting the launch of replacement spacecraft, which would be incorporated into the team, which in turn would initiate necessary self-configuration and self-optimization.

Individual ANTS spacecraft will have self-healing capabilities also. For example, an individual may have the capability of detecting corrupted code (software), causing it to request a copy of the affected software from another individual in the team, enabling the corrupted spacecraft to restore itself to a known operational state.

Self-optimizing Optimization of ANTS is performed at the individual level as well as at the system level.

Optimization at the ruler level is primarily through learning. Over time, rulers will collect data on different types of asteroids and will be able to determine which asteroids are of interest, and which are too difficult to orbit or collect data from. This provides optimization in that the system will not waste time on asteroids that are not of interest, or endanger spacecraft examining asteroids that are too dangerous to orbit.

Optimization for messengers is achieved through positioning, in that messengers may constantly adjust their positioning in order to provide reliable communications between rulers and workers, as well as with mission control back on Earth.

Optimization at the worker level is again achieved through learning, as workers may automatically skip over asteroids that it can determine will not be of interest.

Self-protecting The significant causes of failure in ANTS will be collisions (with both asteroids and other spacecraft), and solar storms.

Collision avoidance through maneuvering is a major challenge for the ANTS mission, and is still under development. Clearly there will be opportunity for individual ANTS spacecraft to coordinate with other spacecraft to adjust their orbits and trajectories as appropriate. Avoiding asteroids is a more significant problem due to the highly dynamic trajectories of the objects in the asteroid belt. Significant planning will be required to avoid putting spacecraft in the path of asteroids and other spacecraft.

In addition, charged particles from solar storms could subject spacecraft to degradation of sensors and electronic components. The increased solar wind from solar storms could also affect the orbits and trajectories of the ANTS individuals and thereby could jeopardize the mission. One possible self-protection mechanism would involve a capability of the ruler to receive a warning message from the mission control center on Earth. An alternative mechanism would be to provide the ruler with a solar storm sensing capability through on-board, direct observation of the solar disk. When the ruler recognizes that a solar storm threat exists, the ruler would invoke its goal to protect the mission from harm from the effects of the solar storm, and issue instructions for each spacecraft to "fold" the solar sail (panel) is uses to charge its power sources.

Self-aware Clearly, the above properties require the ANTS mission to be both aware of its environment and self-aware.

The system must be aware of the positions and trajectories of other spacecraft in the mission, of positions of asteroids and their trajectories, as well as of the status of instruments and solar sails.

3.2 Real time issues

The swarm-based concepts of ANTS (or its submission, Prospecting Asteroid Mission, PAM, as described above) enable exploration missions that never before would be possible. Such concept missions are clearly real-time systems. ANTS must be survivable in a harsh environment (space) over multiple years. The mission must be able to protect itself and to recover from collisions, threats from solar storms, and other problematic issues.

This must all be considered in the context of significant transmission delays. Round-trip delays between Earth and the mission exceed 40 min. The result is that exceptional events cannot be dealt with from Earth. Even anticipated events cannot be dealt with from Earth, as catastrophic damage could have occurred before ground control had even received notification.

The result is that the system *must* be self-managing. In order for its real-time behavior to be realized, the mission *must* exhibit the properties of an Autonomic System, which (as we pointed out in Sect. 2.2) are desirable properties of a RTS in any case.

4 Conclusion

What is clear, is that applications based on such paradigms as we have described, and many envisioned for the future (and certainly not limited to the telecommunications, aerospace or space exploration domain) will be far too complex for humans to address all issues. Moreover, many issues will not be foreseeable, and much behavior will require hard real-time deadlines that can never be met with more traditional approaches.

While there *is* an overhead to achieving autonomicity, we believe that this overhead comes with significant benefit for RTS. We do not believe that it is too "costly"[1] for real-time systems, but rather that in the future it will prove to be essential for developing effective RTS. Moreover, there are techniques that may help to mitigate that overhead and reduce the number of signals that need to be sent.

Simultaneously, Autonomic Computing, and related areas, draw on much of the excellent research produced by the RTS community, a significant proportion of which was essential in making the AC initiative feasible.

In short, we believe that we are moving swiftly towards a time when it will be imperative to have self-managing Real-Time Systems.

Acknowledgments This work was supported in part by Science Foundation Ireland grant 03/CE2/I303_1 to Lero-the Irish Software Engineering Research Centre (www.lero.ie). This research is partly supported at University of Ulster by the Computer Science Research Institute (CSRI) and the Centre for Software Process Technologies (CSPT) which was funded by Invest NI through the Centres of Excellence Programme, under the EU Peace II initiative. This paper is based substantially on a keynote address [16]. Several of the technologies described in this paper were developed with NASA and are patent-pending and assigned to the United States Government.

References

1. Hinchey MG, Sterritt R (2006) Self-managing software. Computer 39(2):107–109
2. Horn P (2001) Autonomic computing: IBM perspective on the state of information technology. IBM T.J. Watson Labs, NY
3. Sterritt R, Hinchey MG (2005) Why computer-based systems should be autonomic. In: Proceedings of 12th annual IEEE international conference and workshop on the engineering of computer based systems (ECBS 2005), 3–8 April 2005, Greenbelt, MD, USA, pp 406–414
4. Kephart JO, Chess DM (2003) The vision of autonomic computing. Computer 36(1):41–52
5. Sterritt R (2005) Autonomic computing. Innovations in systems and software engineering, vol 1(1). ISSN 1614–5046, Springer, Berlin, pp 79–88
6. Sterritt R, Hinchey MG (2005) Birds of a feather session: "autonomic computing: panacea or poppycock?". In: Proceedings of IEEE workshop on the engineering of autonomic systems (EASe 2005) at 12th annual IEEE international conference and workshop on the engineering of computer based systems (ECBS 2005), 3–8 April 2005, Greenbelt, MD, USA, pp 335–341
7. Sterritt R, Bustard DW (2003) Autonomic computing: a means of achieving dependability? In: Proceedings of IEEE international conference on the engineering of computer based systems (ECBS'03), 7–11 April 2003, Huntsville, AL, USA, pp 247–251
8. Sterritt R (2003) Pulse monitoring: extending the health-check for the autonomic GRID. In: Proceedings of IEEE workshop on autonomic computing principles and architectures (AUCOPA 2003) at INDIN 2003, 22–23 August 2003, Banff, AB, Canada, pp 433–440
9. Sterritt R (2002) Towards autonomic computing: effective event management. In: Proceedings of 27th annual IEEE/NASA software engineering workshop (SEW), 3–5 December 2002, Maryland, USA. IEEE Computer Society Press, pp 40–47
10. Sterritt R, Bantz DF (2004) PAC-MEN: personal autonomic computing monitoring environments. In: Proceedings of IEEE DEXA 2004 workshops—2nd international workshop on self-adaptive and autonomic computing systems (SAACS 04), 30 August–3 September 2004. Zaragoza, Spain
11. Sterritt R, Bustard DW (2003) Towards an autonomic computing environment. In: Proceedings of IEEE DEXA 2003 workshops—1st international workshop on autonomic computing systems, 1–5 September 2003, Prague, Czech Republic, pp 694–698
12. Sterritt R, Hinchey MG (2005) From here to autonomicity: self-managing agents and the biological metaphors that inspire them. In: Proceedings of integrated design and process technology symposium (IDPT 2005), 13–17 June, Beijing, China, pp 143–150

[1] We mean in non-financial terms.

13. Lewandowski SM, Van Hook DJ, O'Leary GC, Haines JW, Rossey LM (2001) SARA: survivable autonomic response architecture. In: Proceedings of the DARPA information survivability conference and exposition II, vol 1. June 2001, pp 77–88
14. Truszkowski WF, Hinchey MG, Rash JL, Rouff CA (2006) Autonomous and autonomic systems: a paradigm for future space exploration missions. IEEE Trans Syst Man Cybern C
15. Truszkowski WF, Rash JL, Rouff CA, Hinchey MG (2004) Asteroid exploration with autonomic systems. In: Proceedings 11th IEEE international conference on engineering computer-based systems (ECBS), workshop on engineering autonomic systems (EASe), 24–27 May 2004, Brno, Czech Republic. IEEE Computer Society Press, pp 484–489
16. Sterritt R, Hinchey M (2008) Towards self-managing real-time systems, keynote address. In: Proceedings of the international workshop on real-time software 2008 (RTS 2008) at international multiconference on computer science and information technology, 20–22 October 2008, Wisla, Poland
17. Sterritt R, Gunning D, Meban A, Henning P (2004) Exploring autonomic options in an unified fault management architecture through reflex reactions via pulse monitoring. In: Proceedings of IEEE workshop on the engineering of autonomic systems (EASe 2004) at the 11th annual IEEE international conference and workshop on the engineering of computer based systems (ECBS 2004), 24–27 May 2004, Brno, Czech Republic, pp 449–455